VOLTDB

APPLICATION BRIEF

PARTITIONING

Without parallelism, the performance of a software system is limited to the speed of a single CPU. In recent years, advances in single-CPU performance have slowed, disproportionate to the increased number of cores-per-machine now packed into commodity servers. Hardware design dictates that, to achieve higher performance, a workload must be parallel.

Modern hardware impacts database design

There are two primary ways that modern databases achieve parallelism:

- Threading with locks and shared memory
- Dividing the data into logical partitions

In the first scenario, multiple user operations are performed concurrently, but if two operations require the same data, they must serialize. Threading clearly favors read-heavy workloads.

The second way to achieve parallelism is to logically partition application data. Here, work is divided based on the data the work must access. If two operations require the same logical data, they still must serialize, but many contentious global structures, such as indices, can be split with the partitions to improve overall throughput and achieve orders-of-magnitude higher performance on write-heavy workloads. Partitioning has informed the conception and design of VoltDB's brand of in-memory, NewSQL database.

Today's scale-out database systems generally use a hybrid approach: data is horizontally partitioned across machines in a cluster, while parallelism within a machine is achieved through threading with locks and shared memory. The nascent market for NoSQL databases reflects this demand.

VoltDB and the advantages of data partitioning

VoltDB achieves concurrency purely through data partitioning. The system treats CPU cores as individual members of a cluster and places logical partitions of data at each CPU. Messaging between partitions is abstracted so it works the same way across the network or across the local bus. This approach has several major advantages, and some tradeoffs as well.

First, it scales well for clusters with many machines, for smaller clusters with many cores per machine, or for larger many-core clusters. It is especially advantageous for write-heavy workloads, which are difficult to support with typical caching architectures. Removing the contention and building orderly queues of work, rather than choke points, is a key part of what allows VoltDB to run one or two orders of magnitude faster than other systems on the same hardware.

More importantly, partitioning at the CPU level is architecturally simpler and produces a more predictable performance profile. For developers using VoltDB, all of the data management and manipulation is done in lock-free, single-threaded code. Besides speed, this makes the code more reliable and easier to troubleshoot. Embracing partitioning, even on a single machine, means that apps are treated the same way no matter how big they grow. Most database systems react to growth and behave unpredictably as they scale across many machines. With VoltDB, an app that works well on a quad-core laptop is likely to work well on a 20-node cluster.

NewSQL designed to address specific use cases

As for tradeoffs, this approach to partitioning is more suited to an in-memory architecture and can be difficult to meld with external transaction control. It works poorly with long-running transactions that can block partitions of data for extended periods. The designers of VoltDB made a choice to target writeheavy, high-throughput applications for OLTP, live analytics and decisioning, and high-velocity data ingestion. These tradeoffs are not major problems, as they would have been with OLAP or other technologies.

Logically partitioning data is an excellent way to scale applications on both clusters of commodity servers, as well as modern many-core systems. Partitioning at the CPU level enables VoltDB to support many-core scaling and incredible throughput, all in a simple, focused product.

209 Burlington Road, Suite 203
Bedford, MA 01730
Phone: +1.978.528.4660
Fax: +1.978.528.0568
http://voltdb.com

VOLTDB